НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри

_____ О.В.Коваль
(підпис) (ініціали, прізвище)

“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 121 Інженерія програмного забезпечення

на тему АРМ енергоменеджера з обліку поточних активів

Виконав : студент 4 курсу, групи ТІ-61
Попадинець_Андрій_Олександрович

_____
(підпис)

Керівник :
к.ф.м.н,_доцент_Тарнавський_Юрій_Адамович _____
Рецензент :
к.т.н,_доцент_Новаківський_Євген_Валерійович _____

(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки: 121 Інженерія програмного забезпечення
Спеціалізація: Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р

.

ЗАВДАННЯ
на дипломну роботу студенту

__Попадинцю_Андрію_Олександровичу_____

(прізвище, ім’я, по батькові)

1. Тема роботи АРМ_енергоменеджера_з_обліку_поточних_активів____
керівник роботи к.ф.м.н, доцент Тарнавський Юрій Адамович
затверджена наказом вищого навчального закладу від ”25” травня 2020р.
№ _____

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Javascript, React, MVC.
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Порівняльний аналіз існуючих рішень, вибір методів вирішення задачі, проектування системи, розробка програмного продукту.
5. Перелік ілюстративного матеріалу
_____
_____
_____
_____
_____

7. Дата видачі завдання ”___”_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	18.02.2020	
2.	Вивчення та аналіз задачі	19.02.2020 – 05.03.2020	
3.	Розробка архітектури та загальної структури системи	06.03.2020 – 31.03.2020	
4.	Розробка структур окремих підсистем	01.04.2020 – 15.04.2020	
5.	Програмна реалізація системи	16.04.2020 – 15.05.2020	
6.	Оформлення пояснювальної записки	16.04.2020 – 15.05.2020	
7.	Захист програмного продукту	16.05.2020	
8.	Передзахист	11.06.2020	
9.	Захист	17.06.2020	

Студент _____ Попадинець А.О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Тарнавський Ю.А.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

«Автоматизоване робоче місце енергоменеджера з обліку поточних активів». КПІ ім. Ігоря Сікорського, Київ, 2020.

Дипломний проект присвячений розробці універсальної системи, що слугуватиме для моніторингу та обліку енергетичних ресурсів, автооновленню даних з лічильників та обробкою інформації у вигляді графічних таблиць для виведення на екран.

Текстова документація містить інформацію про розроблені алгоритми, застосовані шаблони проектування, розробку та тестування. Також в документації детально описаний обраний для реалізації проекту стек технологій.

Ключові слова: АРМ, енергоменеджер, енергетика, експлуатація, React, Laravel, Javascript , MVC.

Розмір пояснювальної записки – 47 аркушів, містить 20 ілюстрацій.

ABSTRACT

"The workstation of energy manager system for asset accounting.." NTUU KPI Igor Sikorsky, Kyiv, 2020.

The diploma project is devoted to the development of a system that will serve to asset accounting of energy consumption for improving and researching ways to decrease energy expenditure.

The text documentation contains information about the developed algorithms, applied design template, development and testing. The documentation also describes in detail the technology stack selected for project implementation.

Keywords: management, energy, asset accounting, Laravel, React, Javascript, MVC.

The size of the explanatory note - 47 sheets, contains 20 illustrations.

Аннотация

«Автоматизированое рабочее место энергоменеджер по учету текущих активов». КПИ им. И. Сикорского, Киев, 2020.

Дипломный проект посвящается разработке универсальной системы, что служит для учета текузих энергетических ресурсов, автообновлению данных с счетчиков и обработкой информации в виде графиков для выведения на экран.

Текстовая документация содержит информацию о разработанных алгоритмах, применяемых шаблонах прооектирования, разработку и тестирования. Также в документации подробно описан избраный для реализации проекту стек технологий.

Ключевые слова: АРМ, энергоменеджер, энергетика, эксплуатация, React, Laravel, Javascript , MVC.

Размер записки – 47 страниц, включает в себя 20 иллюстраций.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

1. БД- база даних
2. СУБД – система управлінням базами данних
3. АРМ – Автоматизоване робоче місце.
4. DOM - Об'єктна модель документа
5. JS – JavaScript
6. СКВ – система контролю версій
7. MVC – архітектура розробки програмних засобів model – view - controller
8. ІБ – інформаціна база
9. ОС – операційна система
10. ID – унікальний ідентифікатор

ВСТУП

Наразі питання управління та використання енергетичних ресусів є одним з основних та найпоширеніших питань для обговорення по всьому світі. Проблеми екології, неправильне використання ресурсів та спроби досягнення максимальної продуктивності з мінімальною затратою ресурсів.

З розвитком технологій людство зрозуміло важливість автоматизації механічних процесів та процедур для збільшення ефективності виробництва та заощадження коштів. Сфера енергоменеджменту не є виключенням – все частіше повсякденні речі, які раніше робила людина, починають робити автоматизовані машини. Швидкі та точні розрахунки, облік величезної кількості матеріалу, миттєві виведення даних для аналізу у вигляді графіків-це одні з основних принципів заощадження цінних енергоресурсів.

Впровадження систем енергомоніторингу на основы автоматизованого робочого місця є базовою функціональною складовою у впровадженні системи енергоменеджменту. Аналізуючи інформацію із мережі Інтернет, стає зрозуміло що наразі фактично не існує універсальних внутрішніх систем або ж платформ, які б дозволили розробити систему енергоменеджемнту та обліку активів під власні потреби.

Подібні системи, що створюються на підприємствах для обліку енергоресурсів, створені на комерційній основі та, фактично, не доступні більшій кількості користувачів світової павутини WEB.

Саме тому було вирішено створити власну систему обліку та контрою енергоресурсів – систему енергоменеджменту.

Було вирішено розробити систему, що дозволить обробляти данні та зберігати інформацію про використання енергоресурсів, та температурних режимів, з метою оптимізації їх використання.

Головні принципи системи АРМ енергоменеджера повинні базуватись на концепціях надійності, швидкості та простоті у використанні.

Записка складається із 5 розділів.

Перший розділ містить опис задачі розробки системи, другий описує предменту область та проблематику задачі. Третій розділ описує засоби розробки, що були використані у роботі над проектом. Четвертий розділ містить опис реалізації продукту та архітектуру програми, п’ятий розділ містить опис роботи користувача із системою

1. ЗАДАЧА РОЗРОБКИ СИСТЕМИ АРМ ЕНЕРГОМЕНЕДЖМЕНТУ

Система АРМ енергоменеджменту автоматизує обробку вхідних даних та аналізує використання енергоресурсів, також присутній моніторинг температурних режимів будівель. Головна задача – покращити ефективність використання енергоресурсів та надати змогу аналізувати температурні режими в корпусах та складських приміщеннях “КПІ”’, Це дозволить швидко виправляти недоліки у енергоефективності будівель та спостерігати за коректним дотриманням температурних норм.

Система може використовуватися службою енергоменеджменту НТУУ «КПІ» : головним енергоменеджером, та відповідальними особами у кожному корпусі. Головний енергоменеджер має можливість додавати користувачів із певними правами доступу до системи.

1.1. Потенційні користувачі системи

Система може використовуватися службою енергоменеджменту НТУУ «КПІ» : головним енергоменеджером, та відповідальними особами у кожному корпусі. Головний енергоменеджер має можливість додавати користувачів із певними правами доступу до системи. Відповідно до рівня

доступу, користувач може додавати та/або редагувати дані для обліку активів в системі.

1.2. Загальна характеристика функціоналу системи

Згідно до основних потреб служби енергоменеджменту, стояла задача розробити веб-систему що буде містити інформацію про: корпуси та аудиторії. Повинні бути описані головні характеристики будівель. Основна статична інформація про площу, рік побудови, рік останніх реновацій в будівлі, інформація про лічильники(номер, серія, тощо), а також динамічна(показники лічильника).

Відповідальна особа може також вносити інформацію про температури у аудиторіях, отримувати інформацію про температури. Платформа, також, має можливість експортування данних у Excel-таблиці.

Головний енергоменеджер-найбільш привілейована модель сторінки користувача та може додавати нових користувачів, та змінювати їм права доступу.

1.3. Особливі сторони розробки системи АРМ

Для максимальної продуктивності та комфорту роботу з системою було розбито на підсистеми, згідно принципу клієнт-серверної архітектури. Як першочергове завдання було вирішено поставити задачу розробки

серверної частини додатку, та передбачення взаємодії через application programming interface (API).

За допомогою клієнт-серверної архітектури з’являється змога розбити програму на логічні компоненти, які нічого один про одного не знають: сервер зберігає та обробляє вхдіні дані, клієнт – відображає їх у зрозумілому для користувача вигляді.

Клієнт-серверна архітектура забезпечує надійність на безпеку при роботі з даними.

2.1. Аналіз проблеми енергоменеджменту.

Було проведено детальний аналіз проблеми використання енергетичних ресурсів та питання раціонального управління енергоменеджменту. Проаналізовано можливі вирішення проблеми та їх актуальність.

2.1.1. Проблема використання енергетичних ресурсів

Проблематика неефективних витрат енергетичних ресурсів на протязі довгих років скупчувалась в одну з найважливіших загальнолюдських проблем. Коректне та економічно вигідне використання природних енергоресурсів, зменшення шкідливих парників в атмосферу та правильно-розподілене використання електро та тепло енергії набувають надзвичайно важливого значення у сьогоденному соціумі.

2.1.2. Загальне поняття енергоменеджменту в Україні та світі

Оскільки Україна входить у список енергодефіцитних країн та задовільняє свої паливні та енергетичні потреби в основному не більше ніж на 50 відсотків, питання ефективного використання ресурсів займає важливу позицію як в економічному так і соціальному плані.

Енергетичний менеджмент – діяльність, що спрямована на забезпечення правильного спрямування корисних ресурсів на рівні підприємства або держави, що дозволяє раціонально оптимізувати обсяг енерговитрат.

Так як інтегральним показником розвитку економіки є ефективне використання енергоресурсів, за цим показником Україна займає позицію числа держав, де стагнація існуючого положення з легким поштовхом провокує економічну кризу.

Головні позиції енергоменеджменту включають в себе:

- моніторинг енергоспоживання;
- обробка існуючих показників як основа складання нових бюджетів;
- розроблення нових маловідходних та безвідходних технологій;
- розробку енергетичних бюджетів;
- розроблення енергетичної політики;
- планування нових енергозберігаючих заходів;
- розроблення енергоефективних систем та засобів контролю та управління за енергоспоживанням та захисту довкілля від забруднення;
- організація загального енергетичного та економічного менеджменту.

Будь-яка система енергоменеджменту в світі включає в себе поняття енергомоніторингу.

Енергомоніторинг включає в себе можливість відтворення цифрових даних про стан технічного обладнання та можливості експлуатації.

Як показує досвід сучасних міст у розвинених країнах – одної з головних допоміжних моделей ефективного керування ресурсами є система енергомоніторингу. Ця система дозволяє заощаджувати споживання енергії до 10, а у деяких випадках навіть до 13 відсотків впродовж дня.

Відомо, шо ,безвиключно, будь-яка країна світу бажає скоротити витрати на імпорт енергоресурсів та посприяти на добування власної сировини, проте це потребує мільярдних бюджетних капіталовкладень, до прикладу:

Експерти прогнозують сценарій сталого розвитку низьковуглецевих джерел, які, в свою чергу подвоять свою частку в структурі енергетики

світу в 2040 році, . Підвищення енергоефективності знизить потребу в зростанні видобутку та виробництва енергії, а без підвищення енергоефективності обсяги кінцевого споживання мали б зростати більш ніж у двічі

Тому можна зробити висновок, що задана система по контролю енергоресурсів (система енергоменеджменту) є корисною та необхідною операційною можливістю на підприємствах для контролю обігу енергетичних одиниць, а також не менш важливою для обліку активів на державному рівні.

2.1.3. Проаналізовані наявні системи енергоменеджменту.

Очевидно що попит формує пропозицію. Тому, не дивно що з збільшенням цікавості до енергетичної промисловості країни чи то навіть підприємства, стало питання розробки унікальних систем для обліку та контролю ресурсів на різних рівнях.

Запропоновані системи пропонують спектр модулів відповідно до вимог користувача. До прикладу:

2.1.4.Автоматизована система енергомоніторингу.

Автоматизована система енергомоніторингу (АСЕМ) представляє собою систему програмного та технічного постачання для віддаленого обліку та контролю споживання енергетичних ресурсів. АСЕМ є багаторівневою системою та забезпечує автоматизований обрахунок енеретичних ресурсів на основі інформації, що отримана відповідно до вузлів обліку тепла, електричної енергії, постачання водних реурсів, а також збір інформації про екстраординарні(аварійні) сигнали та температуру повітря як всередині так і зовні приміщень.

Основним завданням створення АСЕМ стала проблема основної та точної оперативно отримуваної інформації щодо питань контролю, а також підвищення енергоефективності споживчих факторів та раціонального розпорядження енергетичних ресурсів будівель.

Рис. 2.1 АСЕМ. Загальна інформація про будівлю

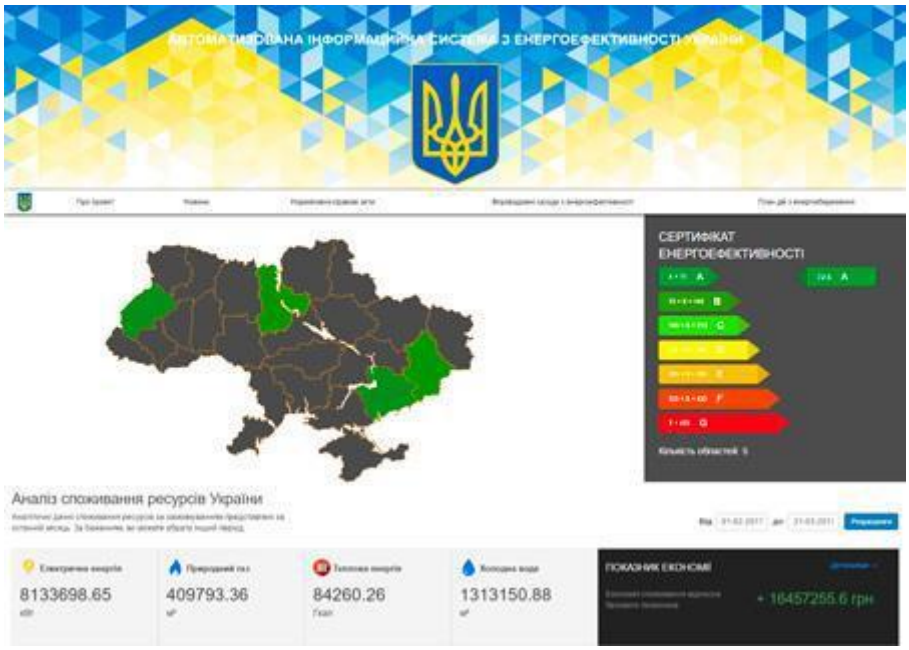
Основні модулі та функціональні можливості АСЕМ

1. Автоматизований облік та контроль даних енерговикористання будівлі.
2. Передача даних обліку щодо розмірів спожитих енергетичних ресурсів для постачальників
3. Інформаційні повідомлення щодо неприємних нештатних ситуацій
4. Підготовка звітності для розробки послідовних правильних дій
5. Прогнозування розходу енергетичних ресурсів у різних можливих сценаріях
6. Захист даних від несанкціонованого доступу.

2.1.5.АІС Енегросервіс

Система АІС - це платформа для щоденного контролю енергоспоживанняспоживання в бюджетних та приватних установах, головною ціллю якої є контролювання та аналізування щодо споживаємих ресурсів у розрізі кожного об'єкту (будинку), де впроваджено систему, з доступом контролю примітивного рівня користувача(споживача), лімітів та інших контрольних параметрів, прогнозування використання або економії ресурсів, контроль фактичного заощадження та інших необхідних заходів. Окрім цього, АІС представляє собою платформу для формування ресурсозбережливої філософії, розміщення даних про можливості та гілки ресурсозбереження, джерела фінансування енергозберігаючих засобів та кращих практичних навичок.

Рис. 2.2. Головна сторінка АІС Енергосервіс.



Система розподілена на модулі:

1. Модуль об’єкту
2. Модуль обліку
3. Модуль аналізу
4. Модуль контролю регламенту та оповіщення
5. Модуль адміністрування

Ці модулі не є налаштовуваними, тому користувачу залишається лише користуватись заданими можливостями системи.

2.1.6.Поняття енергоефективності.

Енергетичний менеджмент – це сфера, яка направлена на визначення та забезпечення раціонального користування енергетичними ресурсами на муніципальному та підприємницькому рівні, та забезпечує оптимізацію деякого обсягу витрат енергетичних ресурсів.

Найкращий спосіб досягнути енергоефективності завжди бере початок з енергоменеджменту. Це поняття певного інструменту, який дає можливість оптимізовувати енергетичні витрати в будівлях, а також зменшити енергоспоживання без значних капіталовкладень.

Загальносвітова практика делегує про зріст енергоефективності який може досягатися, в більшості, за рахунок змін у організації в системі

менеджменту енергогосподарством підприємства або міста. Впроваджуючи платформу енергетичного керування є здатність без великих фінансових затрат досягнути збільшеного заощадження енергії приблизно на 3-5% за термін у 1-2 роки.

Енергоменеджмент включає в себе спектр заходів, ціль яких, в основному, зосереджена на заощадження енергетичних ресурсів: облік енергоспоживання, розробку енергетичних бюджетів, контроль існуючих показників як бази створення нових бюджетів, а також розробки енергетичної політики, формування нових енергозберігаючих способів.

2.1.7. Висновок

У розділі було досліджено поняття систем енергомоніторингу, та поняття енергоменеджменту, досліджено призначення систем та актуальність їх використання. Наведено приклади використання систем та обгрунтовано ефективність їх використання, а також описані наявні системи енергообліку для наглядного прикладу використання даного програмного засобу.

2.2. Автоматизоване робоче місце. Концепція розробки.

Для розробки подібної системи та забезпечення користувачів системи комфортним автоматичним робочим місцем(далі АРМ) було проведене ретельне дослідження структури та класифікації АРМ.

Засоби АРМ

АРМ поєднує у собі програмно-апаратні засоби, які забезпечують співпрацю користувача з комп'ютерним засобом, а також представляє здатність введення даних (через клавіатуру, комп'ютерну мишку або сканер тощо) і їх виведення на монітор, принтер , звукову карту.

АРМ оператора включається до списку автоматизованої системи керування.

АРМ у системі управління — це проблемно орієнтований комплекс технічних, програмних, лінгвістичних засобів, що встановлений відповідно саме на робочому місці користувача та призначений для автоматизації дій взаємодії людини з комп'ютерної машини в період проектування, а також реалізації завдань.

2.2.1. Призначення АРМ Цілі

АРМ:

- вирішення деякого типу завдань, які поєнані базовою теорією опрацювання інформацією, згуртованістю режимів роботи, що в більшості випадків для спеціалістів економічних установ;
- формалізацію професійних вмінь, тобто здатністю предоставлення за допомогою АРМ самостійно автоматизувати нові модулі та вирішувати нові задачі у процесі накопичення досвіду роботи з системою;
- модульна побудова, яка забезпечує сполучення АРМ з рештою елементів системи аналізу інформації, а також покращення і додавання можливостей АРМ без зупинки його функціонування;
- ергономічність, а саме створення для користувачів комфортні умови роботи та зручного інтерфейсу роботи з системою.

Класифікація АРМ

За рівнем управління

- АРМ вищих керівників — директора, його заступників, головного менеджера, а також управлінців середнього та нижнього рівнів керування;
- АРМ спеціалістів — АРМ аналізуючих осіб, бухгалтерів, економістів; диспетчерів, інженерів та
- АРМ технічних виконавців — бригадирів, майстрів, секретарів тощо.

За рівнем та формою організації праці працівників на персональних комп'ютерах

- Індивідуальні АРМ
- Колективні АРМ

За рівнем використання ПК

- АРМ нижчого рівня
- АРМ середнього рівня
- АРМ вищого рівня

За ступенем підготовленості користувача

- користувачі, які володіють технологіями програмування;
- користувачі, що одержали спеціалізовану підготовку з використовування пристроїв АРМ та освоїли роботу на персональному комп’ютері;
- користувачі, що не отримали спеціальних навичок, проте ознайомлені з певними навичками роботи на конкретному ПК;
- користувачі, що зовсім не мають знань в області персонального комп’ютера та не вміють з нимим працювати.

За видами вирішуваних завдань

- для вирішення інформаційно-обчислювальних задач;
- для вирішення задач підготовки і заведення інфомаційних даних;
- для вирішення інформаційно-довідкових задач;
- для вирішення задач бухгалтерського обліку;
- для вирішення задач статистичної обробки даних;
- для вирішення задач аналітичних розрахунків та інших задач.

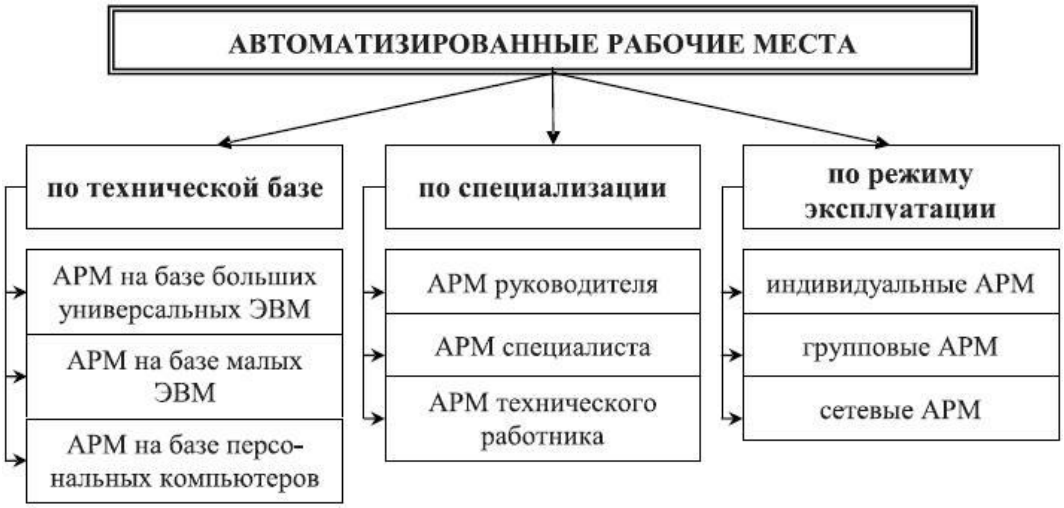
Класифікація АРМ залежить від

- сфери користування (дослідницька діяльність, проектування, менеджмент);
- типу ЕОМ (електронно-обчислювальної машини: мікро-, міні-, макроЕОМ);
- режиму експлуатації (індивідуальний, груповий, мережевий, базовий);

Основні ознаки АРМ

- легкість доступу для користувача до списку технічних, програмних, інформаційних засобів;
- розташування обчислювальних апаратів відповідно на робочому столі користувача;
- здатність створення та покращення проектів автоматизованої обробки інформації в заданій сфері;

Рис. 2.2.1. Опис типів АРМ



2.2.2. Структура АРМ

Структура та склад компонентів будь-якого АРМ напряму залежить від сфери його призначення, типу розбору питань, системи програмного забезпечення, а також способу фіксації даних у початкових документах.

Правильно скомпонована структура – основний компонент успіху автоматизованого робочого місця. При розробці додатку для АРМ потрібно брати до уваги ключові аспекти структуризації для побудування коректної моделі. Основні ключові частини складаються з:

Функціональна частина АРМ є системним важливим компонентом його структури, яка визначає основні моделі фахівця з персоналу, а також процес роботи АРМ у часі, як процес співпраці елементів, що забезпечують безперебійну роботу менеджера. Функціональна частина АРМ включає в себе опис взаємопов'язаних завдань, що включають всі типи формалізованої діяльності робітника. Завдання — це частина функції менеджменту, яка пояснює алгоритм або групу алгоритмів — формування вихідних файлів, які включають деяке функціональне призначення в керувані певним об'єктом.

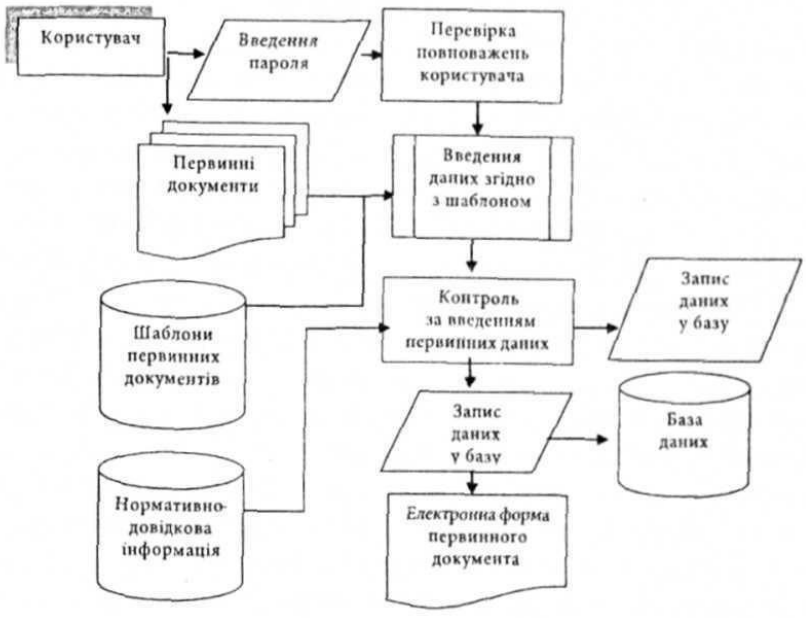
Забезпечуюча частина АРМ — це сукупна складова технічного та інформаційного забезпечення системи.

Технічне забезпечення АРМ — це комплекс технічних засобів, що зроблений на основі персонального комп'ютера(ПЕОМ).

Використання ПЕОМ докорінно змінює технологію та групування процесів управління. З використанням ПЕОМ впроваджуються нові інформаційні технології керуванням персоналом, які базуються на організації АРМ фахівця з кадрової роботи.

Інформаційне забезпечення АРМ — це структура засобів та певних методів конструювання інформаційної бази (ІБ), що розподілена на позамашинне та внутрішньомашинне.

Рис. 2.2.2. Примітивна структура АРМ.



Технологія процесу на АРМ складається з таких етапів

- колекція даних і введення їх у ПК;
- створення інформаційної бази;
- обробка інформації на персональної електронної обчислювальної машини; виведення готової обробленої інформації;
- збереження інформації.

Типова структура АРМ

Будь-яке АРМ повинно містити безвиключно важливі ключові елементи без яких, поняття АРМ, фактично, буде неприпустиме.

Типове АРМ містить:

- транслятори (інтерпретатори) різноманітних мов програмування;
- засоби проектування й обробки даних (редактори текстової, графічної інформації, табличні процесори, генератори вихідних форм);
- програми користувача (обробні, навчальні, СУБД тощо). Розв'язок задач за допомогою АРМ що зв’язані з пошуком необхідної інформації в базі даних та подальшою її обробкою за алгоритмами і видачею результатів на екранний носій чи принтер.

Принципи роботи АРМ

В основу організації системи управління персоналом з використанням АРМ-ів мають бути покладені такі принципи:

- автоматизована обробка облікових даних в реальному часі на робочих місцях фахівців з праці;
- гуртування файлів на носіях, що считаються машинним апаратом;
- групування і виведення результативних даних як запит в необхідному для фахівця з праці обсязі.

Висновок до розділу

АРМ - це діалогова людино-комп’ютерна система, що є організованим продуктивним середовищем по обробці інформації, представлена методичними, організаційно-правовими, лінгвістичними, програмними, технологічними, ергономічними засобами, що забезпечують реалізацію професійних функцій виконавця (керівника, фахівця) конкретної предметної області безпосередньо на його робочому місці.

Було проведено аналіз коректної структури АРМ та групування найнеобхідніших складових компонентів для правильної розробки системи на базі АРМ. При створенні такої платформи потрібно відштовхуватись від принципу її роботи(а точніше специфікації) та області впровадження операційної системи на базі автоматизованого робочого місця.

3. ЗАСОБИ РОЗРОБКИ

Для розробки програмного забезпечення, а саме його клієнтської частини, було обрано наступні технології: мова програмування JavaScript, фреймворк ReactJS, постачальник модулів Webpack, менеджери пакунків Yarn та NPM та Sass — скриптова метамова, яка інтерпретується в каскадні таблиці стилів.

Також було обрано такі засоби розробки: середовище розробки WebStorm, система контролю версій Git, бібліотека для тестування Jest.

Стилізування сайту відбувається за допомогою мови каскадних стилів CSS.

Базовий функціонал зберігання даних розроблений за допомого бібліотеки

React Redux, що дозволяє з меншою затратністю керувати станом компонентів додатку.

Основна програмна частина додатку виконувалась на мові програмування Javascript. бібліотека React, особливість розробки веб-додатку з допомогою цієї бібліотеки полягає у можливості розбити систему на реактивні компоненти, що дає змогу неодноразово використовувати аналогічні модулі на різних рівнях, а унікальність віртуального DOM у бібліотеці дозволяє створювати серйозні проекти які не будуть споживати багато ресурсів.

Відповідно до архітектури програмування систем MVC, а саме розбиття на серверну та клієнтську частини, було вирішено обрати для серверної частини мову програмування PHP, а саме- фреймворк Laravel, MySQL —як систему керування реляційними базами даних. Опис реалізації та застосунку програмних засобів описаний відповідно у наступних підрозділах.

Для системи використовувалась готова серверна частина (API), розроблена студентом групи ТІ-61, Бевзою Дмитром.

3.1. Бібліотека React

React — відкрита JavaScript бібліотека для створення веб-інтерфейсів користувача, задача якої полягає у вирішенні проблеми часткового оновлення змісту веб-сторінки при допомозі віртуального DOM, який найчастіше зустрічається при розробці односторінокових застосунків(компонентів).

Найпопулярніші додатки у яких використовується реакт - Facebook, Instagram. Також, широка спільнота індивідуальних розробників користує дану технологію і багатьох своїх локальних проектах.

React дозволяє розробникам створювати великі веб-застосунки, які використовують матеріали даних, можливість яких полягає у здатності змінююватись з часом, без перезавантаження сторінки. React філософія та основні превілегії полягають в тому, щоб бути швидкими, простими, масштабованими (легко звмінювати свій стан без впливу на додаток). React обробляє лише user(користувацький)-інтерфейс у застосунках. Це відповідає *видові* у шаблоні модель-вид-контролер (MVC), та також має змогу бути використане у поєднанні з іншими JavaScript бібліотеками або в найбільшими фреймворках MVC, до прикладу таких як AngularJS. Angular також може використовуватись з React в основі компонентних надбудов, щоб юрати під контроль частини без user-інтерфейсу побудови веб-застосунків. В основному, бібліотеку інтерфейсу користувача React частіше за все використовують в спектрі з рештою бібліотек що доповнюють його, такими як Redux.

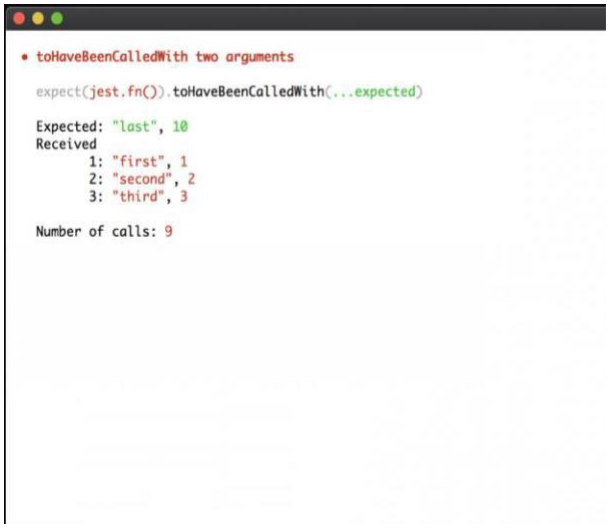
3.2. Бібліоетека для тестування Jest

Jest - це фреймворк для тестування JavaScript коду, що розроблений для запевнення в коректній роботі будь-якого JavaScript коду. Він дозволяє писати мок-тести з прийнятним, загільно-відомим і функціональним API, і швидко досягати бажаних результатів.

Jest може використовуватися в проектах, що використовують Webpack для управління активами, стилями і компіляцією. Webpack вносить деякі унікальні складності в порівнянні з іншими інструментами. Забезпечуючи тестам унікальний глобальний стан, технологія добивається безпомилкового запуску тестів в паралельних потоках. Щоб зробити швидке тестування, Jest в першу чергу запускає провалені тести, а потім черговість їх запуску, відштовхуючись від того, як довго виконується кожен тест.

Приклад моків (Mock:Макети об'єктів у об'єктно-орієнтованому програмуванні-це об'єкти, що імітують поведінку існуючих об'єктів контрольованими способами, тобто, реалізують інтерфейси справжніх об'єктів, але не мають власної реальної функціональності.)

Рис. 3.1 Імітація поведінки об’єкта у бібліотеці Jest



Jest користує призначене для користувача розширювання для імпорту в тестах, що спрощує Mock будь-якого об'єкту поза межами тесту. Є наявним використати імітований імпорт з багатим API Mock Functions для стеження за викликами функцій тестовим синтаксисом.

Дана бібліотека для тестувань була вибрана для подальшого комфортного рефакторингу програмного коду та виявлення нетипової поведінки додатку у певних робочих моментах.

3.3. Мова програмування JavaScript

Для клієнтської частини проекту було обрано мову програмування JavaScript(JS).

JavaScript - це мова програмування, який додає інтерактивність на будь-який веб-сайт (до прикладу: ігри, або ж відгук при натисненні кнопок або при введенні даних у форми, динамічні стилі, анімація).

JavaScript працює зі всіма основнима браузерами, на всіх платформах. Ця мова програмування працює на стороні сервера, зі всіма операціними системами. Будь-яке написання веб-застосунку не обходиться без JavaScript у front-end частині розробки веб-додатку. Доречно було б сказати що не грає ролі на чому розроблена серверна частина (back-end) - будь-то Java, PHP, .NET, Node.js або ж будь-яка інша мова програмування - на стороні клієнта буде необхідний JavaScript.

З часу свого створення до сьогодні, JavaScript завоював серця тисяч програмістів та наразі вважається найбільш розповсюдженою мовою програмування (близько 70 відсотків програмістів хоча б раз за свою кар’єру зустрічались з написанням базових скриптів коду на цій мові), а також з попитом на JS, пішло розмежування та просунення даної мови і на серверну частину також. До прикладу NodeJS, ExpressJS – серверні нащадки

‘ванільного’ JavaScript. Зручність та вміння технології дозволяє ефективно виконувати будь-яку задачу поставлену перед розробником.

3.4. Sass. Скриптова метамова.

Sass (Syntactically Awesome Stylesheets) — це скриптова метамова, яка компілюється в звичайні CSS-стилі. Всі веб-дизайнери, що стикаються з CSS кодом(мова стилів) більше 500 рядків відповідно, мають справу з головним болем на тему того, як би його спростити. На жаль, з часу розробки основних стандартів каскадних стилів їх структура особливо не змінювалася. Відповідно до цієї проблематики було вирішено створити технологію яка не буде в особливості відрізнятись від свого предка, проте буде кардинально спрощувати роботу зі стилізуванням додатків та веб-сторінок, а також зменшувати розмір файлів застосунку.

3.5. Redux

Redux - відкрита JavaScript бібліотека призначена для управління станом програми. В соновному її прийнято використовувати у стеці розробки з React або Angular фреймворками для чіткої побудови інтерфейсів користувача. Redux - це контейнер станів для застосунків JavaScript. Ця технологія вважається одною з головних та найбільш потребованих при складних розробках великих проектів для легкого та зрозумілого написання коду управління стану компонента. За допомогою цієї бібліотеки розробник з легкістю можу оптимізувати код додатку. Окрім того, забезпечує покращення вмінь розробника, до прикладу, обробка коду в поєднанні з його тестуванням, що ,зазвичай, функціонує під час функціонування.

Redux може використовуватись разом із React або іншими бібліотеками. Розмір файлу Redux невеликий, 2kB, включаючи залежності.

3.6. WebStorm

JetBrains WebStorm — інтегроване середовище аналізу та розробки для JavaScript сторінок та веб-додатків від компанії JetBrains, що створене при натхненні від платформи для програмування IntelliJ IDEA. WebStorm є дороблеою спеціалізованою версією платформи PhpStorm,

що пропонує основні та найбільш ефективні його функції. WebStorm завантажується з перед-установленим розширеннями JavaScript (такими як для Node.js), які є в наявності для PhpStorm безоплатно. Dart.

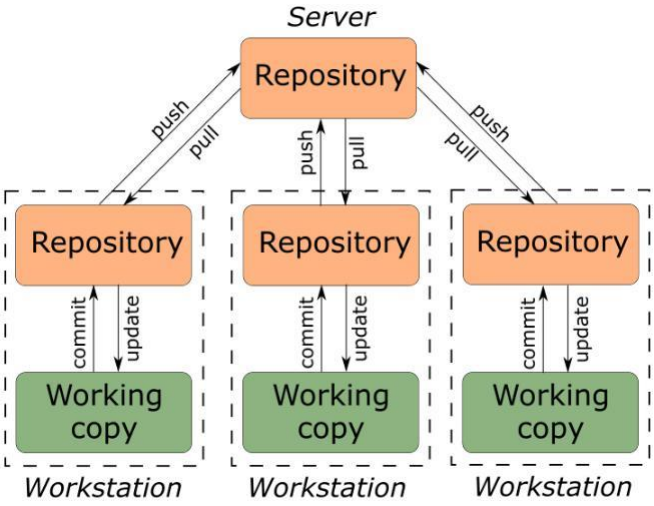
WebStorm подає забезпечення автодоповнення, аналізування коду, легка навігація по проекту, рефакторинг. Найбільшою перевагою інтегрованого середовища програмування WebStorm є співпраця з проектами (у тому числі, рефакторинг JavaScript коду, що включається в різноманітних файлах і папках проекту, а також вкладеного в HTML). Проведена підтримка вкладеності(у файлі на HTML вкладений код на Javascript, у котрий вкладено інший код HTML, всередині якого вкладений Javascript) — в таких моделях є підтримка правильного рефакторингу.

3.7. Система контролю версій GIT

Головною особливістю від решти систем (до прикладу Subversion та подібних цій системі) ми з легуістю можемо вважати те, як Git працює з даними(а саме приймає та обробляє їх). Відповідно до загальноприйнятого досвіду роботи з системами, більшість СКВ утримують дані як список файлових редагувань. Ці системи (до прикладу CVS, Subversion, Perforce, Bazaar) працюють з даними у вигляді списків файлів та змін у кожного з цих списків напротязі деякого проміжку часу (зазвичай це має назву “оснований на дельтах контроль версій”).

Також, у командній розробці система контролю версій дозволяє команді безперешкодно кооперуватися.

Рис. 3.1 Опис системи контролю версій (СКВ)



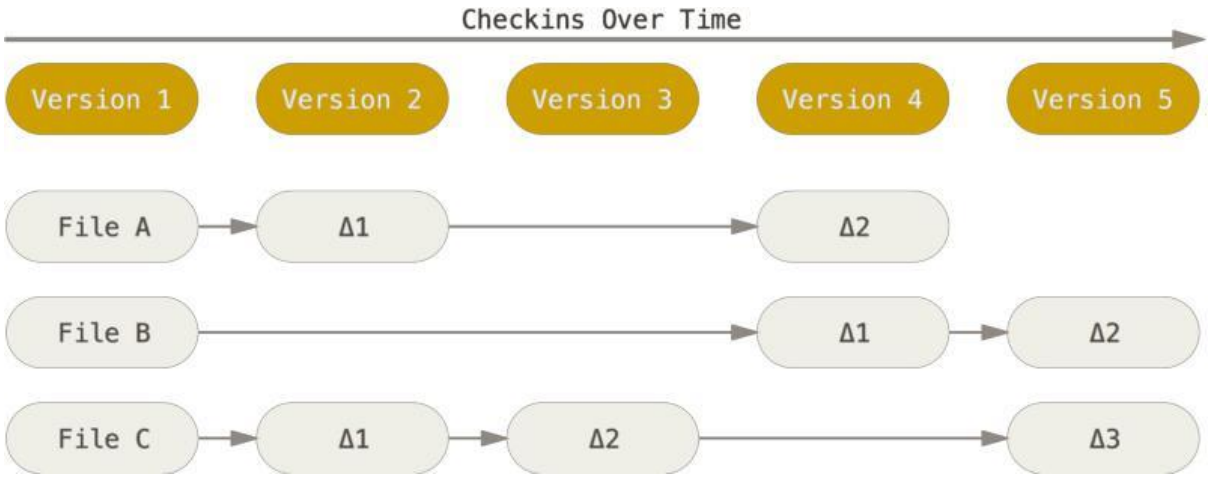
Збереження даних, як деякого списку змін від загальновстановленої примітивної версії кожного файлу.

Git не редегує та не зберігає власну інформацію таким чином. Окрім вищеперерахованого, система Git сприймає свої дані швидше як деяку послідовність скріншотів мініатюрної файлової системи. У Git щоразу, як ви відправляєте запит, тобто зберігаєте стан вашого проекту, Git зберігає інформацію як виглядають всі ваші файли в той момент і зберігає посилання на цей скріншот. Для більшої продуктивності, у випадку коли файли не змінили примітивний вигляд, Git не перезаписує файли, а лише робить посилання на попередній аналогічний документ, який вже зберігається. Git сприймає власні дані більш як потік знімків.

Одже, Git наразі є найпопулярнішою та найбільш ефективною системою контролю версій з існуючих.

Рис. 3.2. Зберігання даних як набору змін відносно

первинної версії кожного з файлів.



3.8. NPM

npm (Node Package Manager) - це менеджер пакунків для мови програмування JavaScript. Для технології виконання аналогічної мови JavaScript - Node.js. Це менеджер пакунків що встановлений по замовчуванню. Він включає в себе клієнт командного рядка, котрий аналогічно називається npm, а в окремості онлайн-базу даних публічних та приватних пакунків, що має назву реєстр npm. Реєстр доступний через клієнт, а доступні пакунки доступні у перегляді та пошуку через веб-сайт npm. Менеджер пакунків та реєстр керуються npm, Inc.

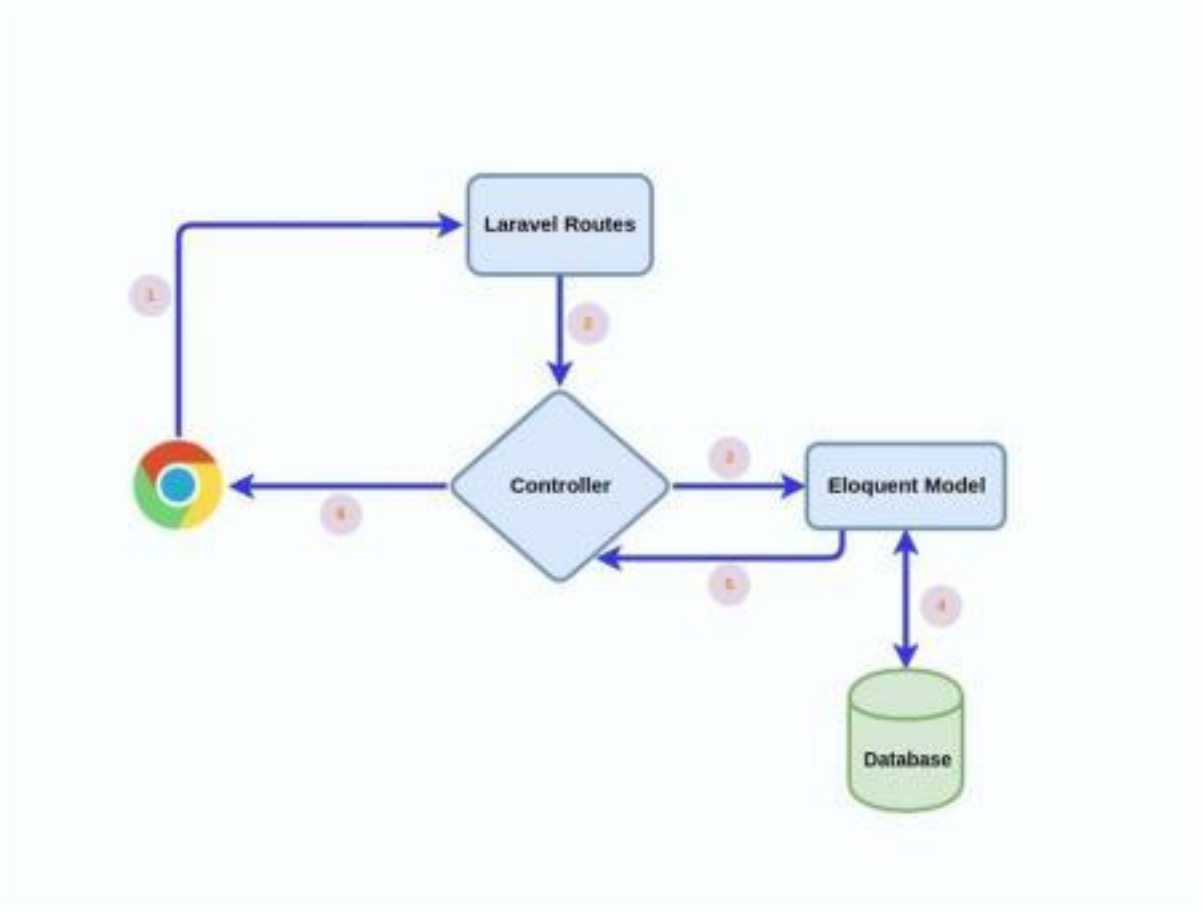
3.8 Серверна частина програмного продукту.

3.8.1. PHP. Laravel.

Для серверної частини було використано мову програмування PHP, а саме- фреймворк Laravel.

Laravel - це фреймворк, який наслідує архітектуру Model- View - Controller (MVC). Кажучи іншими
словами, MVC допомагає занотувати запити до бази даних (Модель) від логіки, пов'язаної з тим, як повинні оброблятися запити (контроллер) і як має бути відображений макет (вид). На зображенні нижче демонструється робота типового MVC Laravel застосування.

Рис. 3.3 Архітектура MVC. Контроллер повертає відповідь.



Роутинг

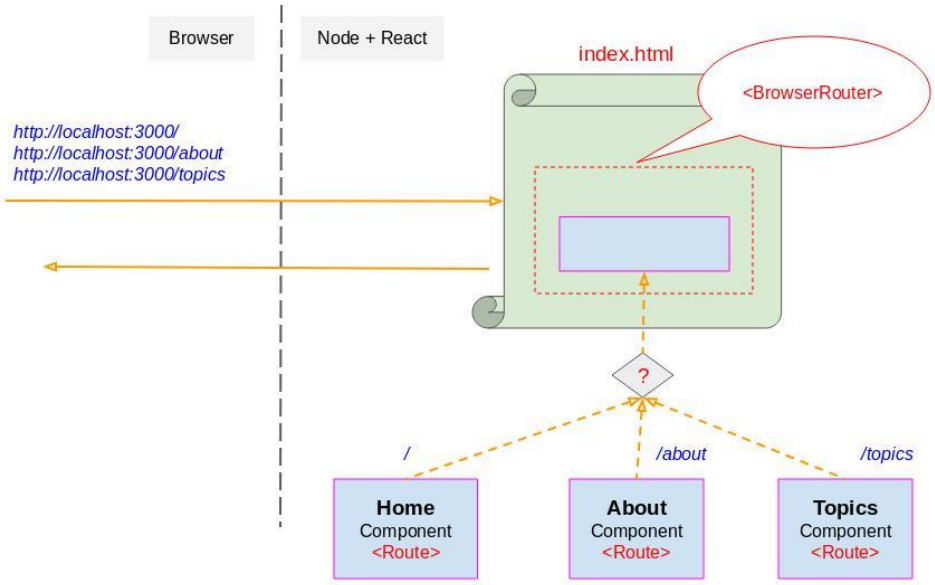
Коли сервер отримує HTTP- запит, Laravel намагається зіставити його з роутом, зареєстрованим усередині любого з файлів роутинга. Усі файли ро утинга знаходяться усередині соответсвующего каталогу. route/web.php мі стить роуты для інтерфейсу, тоді як route/api.php містить роуты для API. Р оуты, зареєстровані в api.php, матимуть префікс /api (як в localhost: 3000/api). Якщо вам треба змінити цю поведінку, ви повинні перейти в кла с RouteServiceProvider в /app/Providers/RouteServiceProvider.php і внести ту ди зміни.

Контролер

У файлі роутинга нині розміщується логіка роутинга і обробки запитів. Ми можемо перемістити логіку обробки запитів в клас контроллера, щоб наш код був краще організований і більше читабельним.

Клас Controller складається з різних методів (index, show save,update,delete) , які відповідають різним HTTP - діям . Я перемістив логіку обробки запитів з роутинга в контроллер.

Рис. 3.4. Загальна схема ReactRouter.



3.9.2 MySQL

Для зберігання даних у додатках використовуються системи керування базою даних.

PHP підтримує роботу з базою даних MySQL.
Спеціальні вбудовані функції для роботи з MySQL дозволяють просто і ефективно працювати з цією СУБД: виконувати будь-які запити, читати і записувати дані, обробляти помилки.
Сценарій, який підключається до БД, виконує запит і показує результат, складатиметься всього з декількох рядків. Для роботи з MySQL не потрібно нічого додатково встановлювати і настроювати; усе необхідне вже доступно разом із стандартним постачанням PHP.

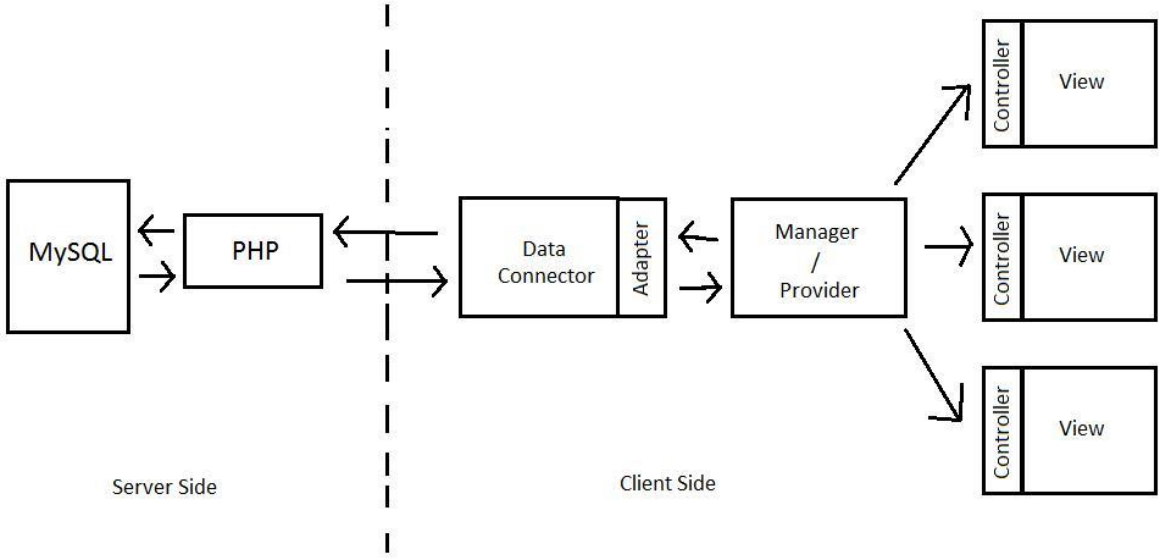
База даних (БД) – упорядкований набір логічно взаємопов’язаних даних. Система керування базою даних (СКБД) - комплекс програмних і мовних засобів, що є необхідними для створення баз даних.

Першочерговим завданням БД є збереження даних і надання доступу до них користувачеві або сторонньому додатку.

Переваги MySQL:

1. Просте використання та встановлення
2. Потужна швидкість
3. Серйозна система безпеки
4. Відкритий код.

Рис. 3.5 Опис роботи з mySql.



4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Згідно до індивідуального завдання, було розроблено клієнтську частину платформи енергоменеджменту, що дозволить користувачу змогу керувати, обновляти та отримувати данні відповідно до рівня доступу сторінки користувача.

Для розробки клієнтської, як було зазначено вище, було обрано мову програмування JavaScript та фреймворк React(ReactJS). Обрані технології дозволили побудувати чітку та зрозумілу web-систему, максимально комфортну для користування, із дотриманням всіх сучасних стандартів та кращих практик розробки. Для кофортної роботи з CSS-стилями використовується Sass-технологія, що дозволяє працювати з стилями та легко змінювати окремі ділянки коду без суттєвого впливу на решту інтерфейсу.

Програмна реалізація продукту проводилась з аналізом загальних потреб та основних положень кожного користувача задіяного у сфері енергоменеджменту. Основні концепції та вимоги що були задіяні при розробці полягають у наступних поняттях:

1. Зручність інтерфейсу для швидкої роботи
2. Можливість імпортувати та експортувати дані
3. Поокреме модульне налаштування прав доступу користувача
4. Надійне розмежування модулів для заповнення та зчитування.
5. Зрозуміла робоча сторінка відображення останньооновленої інформації про ті чи інші енерголічильники.

4.2. Опис роботи з базою даних

У програмному продукті прийнято розділення логіки обробки даних, що в примітивному вигляді реалізовується у наступному вигляді: опісля відсилання запиту ззовні, йде виклик до контролера, що, в своїх методах викликає модель, і починає з нею роботу. Але, даний принцип не зовсім правильний, оскільки, наприклад, в випадку деяких корегувань у логіці зберігання даних, потрібно буде проводити повне редагування проекту. Тому, при розробці бекенду системи було використано патерн «Репозиторій».

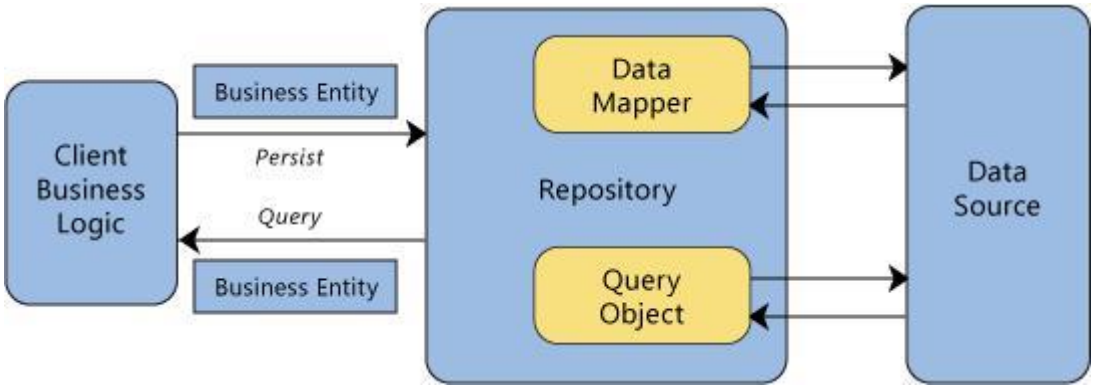


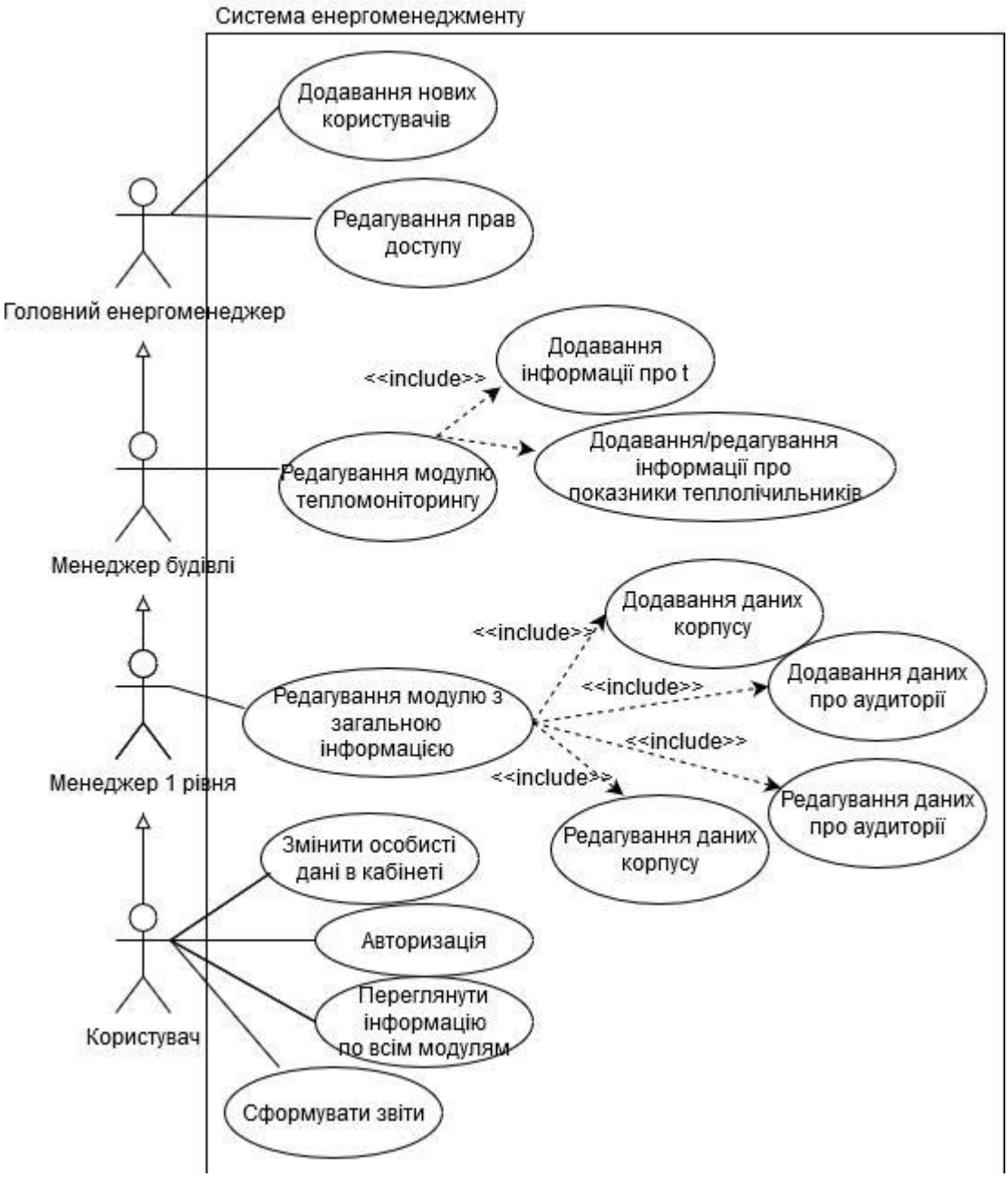
Рис. 4.1. Патерн «Репозиторій»

Даний патерн ховає логіку роботи із базою даних, та класами всередині спеціалізованого класу – репозиторія. А також, що важливо що контролер не є прив’язаним до репозиторію «жорстко» - він також містить у собі інтерфейс.

4.3. Опис функціональних можливостей системи

На рисунку зобрежені актори та досупні їм дії.

Рис. 4.2. Модель бази даних системи



4.4. Ієрархія та можливості користувачів у системі

Згідно до індивідуального завдання, у системі було реалізовано ієрархію ролей. Кожен користувач системи відповідно до рівня свого ієрархічного рівня має розподіл відповідно до своїх ролей. Головний адміністратор - головний енергоменеджер має доступ до модулів надання рівня доступу іншим користувачам.

Програмний продукт, отримуючи запит починає виконувати перевірку, чи користувач володіє відповідним рівнем доступу. Модель перевірки ролей доступу користувача передбачена з гнучкістю для можливого розширення ієрархії ролей та рівнів доступу.

В базі даних ролі список ролей зберігається у спеціальній таблиці.

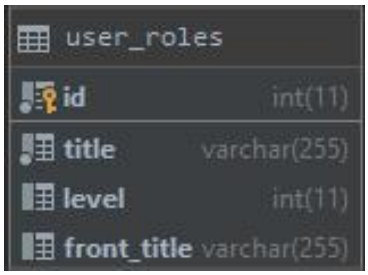


Рис. 4.3 Таблиця з ролями користувачів

В ній зберігається назва ролі, та ID – рівень доступу ролі. Згідно до системи- кожен наступний ієрархічний клас має більше превілегій ніж попередній.

У системі перевіркою ролі користувача займається окремий модуль. Якщо ж рівень ієрархії користувача відповідний до необхідного, дія виконується, якщо ж менший – виводиться попередження про те, що користувачеві не є наявний цей модуль за рівнем доступу.

4.5. Модуль тепломоніторингу

В системі реалізовано модуль тепломоніторингу із наступним функціоналом:

- Заповнення/Редагування/Експортування інформації із тепло лічильників
- Огляд даних із передбачуваних тепло лічильників
- Заповнення/Редагування інформації температурного режиму в аудиторіях
- Огляд даних про температурний режим у аудиторія
- Можливість експортування звітної інформації

Звітна інформація завантажується у форматі файла Excel, і являє собою гнучкий набір фільтрів, за якими є можливість для формування звітних документів.

Переглядати інформацію та завантажувати звіти може будь-який авторизований користувач, але додавати/редагувати/видаляти – лише юзер з відповідним рівнем доступу.

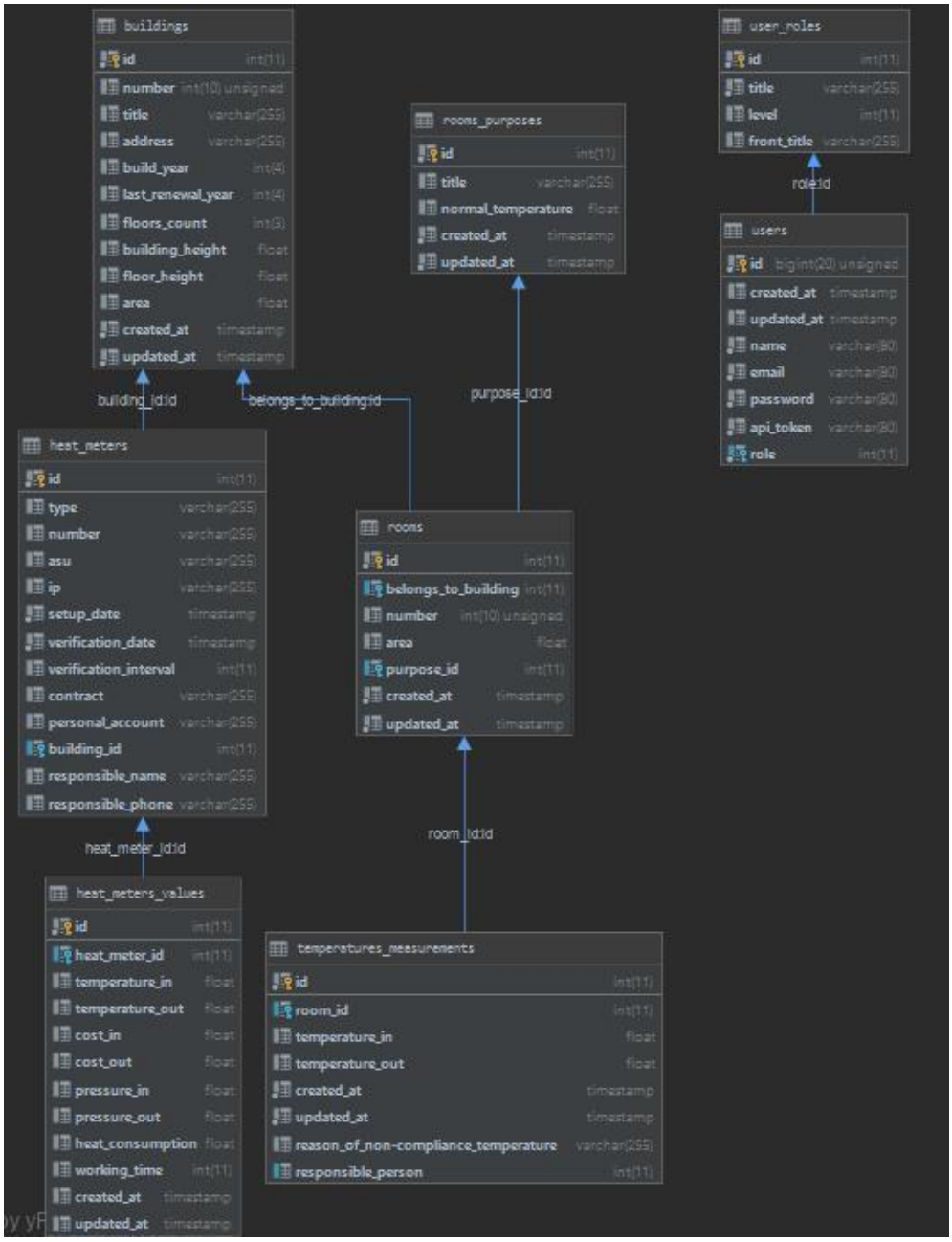
4.6. Особистий кабінет

Список функціональних можливостей користувача у особистому кабінеті відрізняється відповідно до рівня доступу користувача, мінімальний набір становить собою інформацію про дані користувача та вибір базових налаштувань кабінету. Максимальний ж набір дозволяє додавати користувачів та налаштовувати їх профіль відповідно до потреб аккаунту суб’єкта.

4.7. Робота з базою даних.

Для роботи системи, зберігання та зчитування даних було розроблено модель БД, що зображена на рисунку.

Рис. 4.4. База даних проекту.



Згідно до вимог системи, було створено модель БД, яка включає всі потрібні дані та інформацію про корпуси, аудиторії, користувачів та ролі

відповідно. Також, були розроблені таблиці, що включають в себе інформацію про заміряну температуру в кімнатах, а також таблиці з даними теплолічильників – статична інформація про сам теплолічильник, та таблиця із показниками.

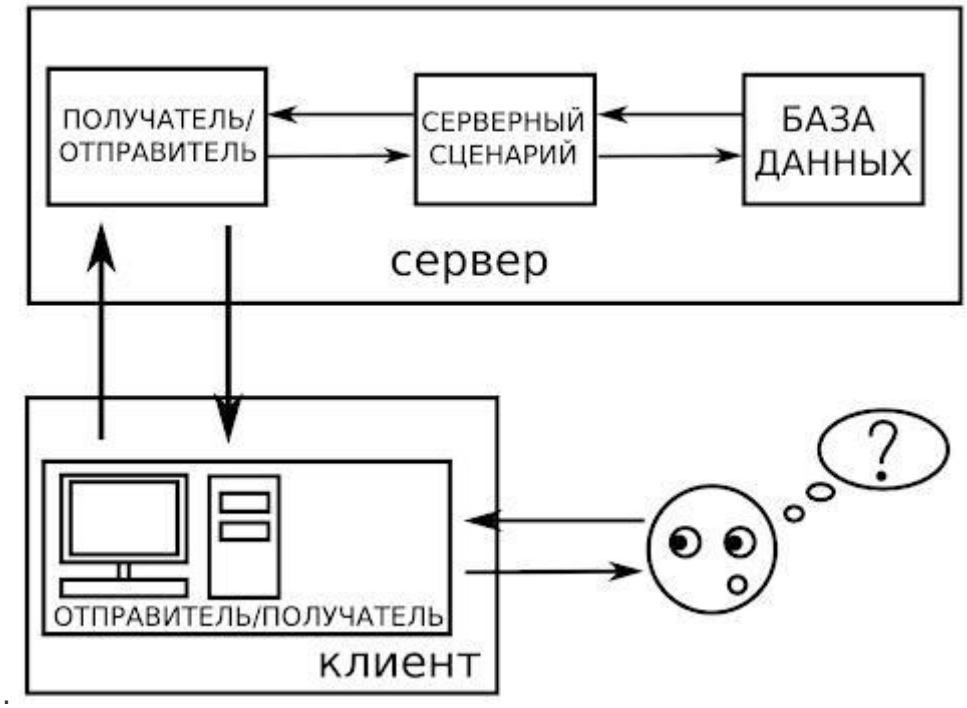
В окрему таблицю було заведено дані про аудиторії та нормативні температури у них.

4.8. Взаэмодія клієнт - сервер

Клієнт-серверна взаємодія один з головних моментів у веб-дизайні. Будь-який сайт, розміщений в мережі Інтернет, грунтується на зв'язці "клієнт-сервер". І це не лише електронні форми. Навіть просте "перегортання" сторінок деякого сайту в Інтернеті - приклад клієнтсерверної взаємодії, адже сторіночки зберігаються не на вашому особистому комп'ютері, а підвантажуються ззовні.

Клієнт - досить широке поняття, починаючи від деякої фізичної особи і закінчуючи деякою програмою на комп'ютері (наприклад, поштовий клієнт). У нашому випадку це комп'ютер, оснащений спеціальним програмним забезпеченням, яке дозволяє користувачеві задати запит до іншої машини і отримати відповідь. Код, що виконується на стороні клієнта, найчастіше називають клієнтським кодом. Він забезпечує створення призначеного для користувача інтерфейсу. Тут, коли мова заходить про браузер, важливу роль грає JavaScript і бібліотеки розширення

Рис. 4.5. Внутрішньосерверна взаємодія

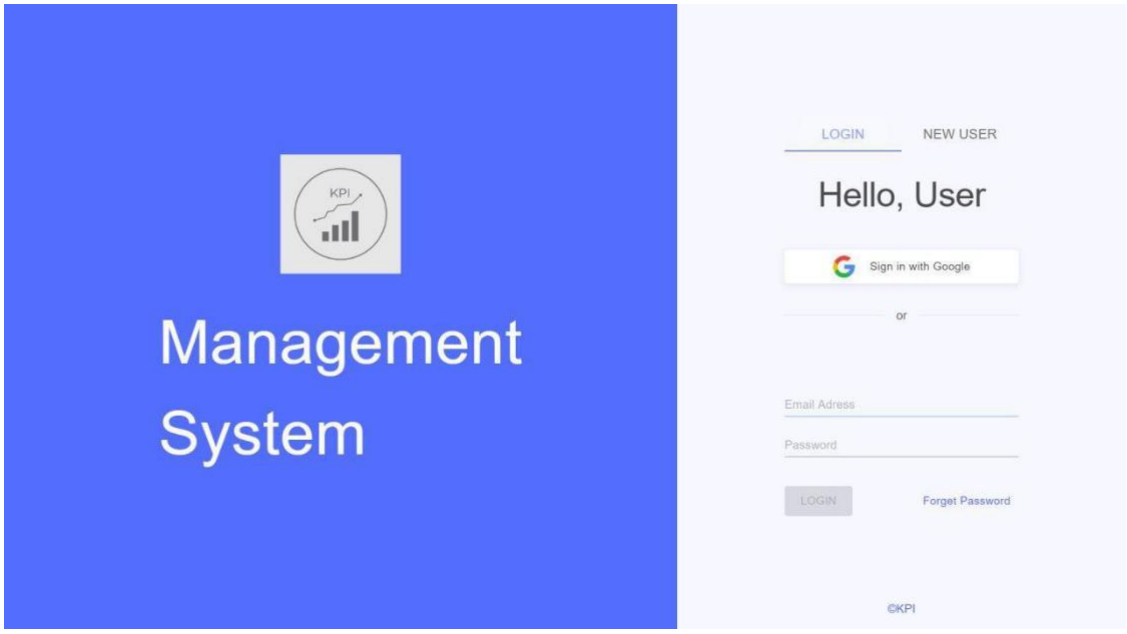


5. ОПИС РОБОТИ КОРИСТУВАЧА ІЗ СИСТЕМОЮ

5.2. Системні вимоги та інсталяція

Виходячи з того що найбільш комфортним варіантом користування системою було б зробити її у вигляді веб-додатку, будь-який користувач з доступом в мережу ІНТЕРНЕТ та браузером (для прикладу Chrome, Firefox, Internet Explorer), має доступ до системи енергомоніторингу.

5.3. Сценарій роботи користувача із системою Рис. 5.1. Сторінка авторизації в систему

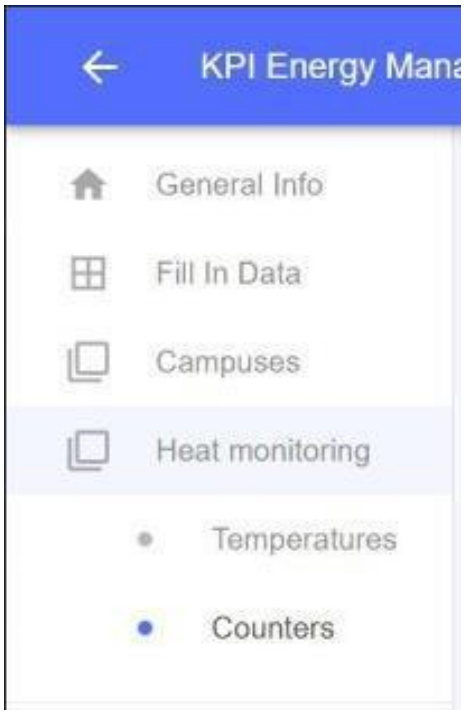


Авторизація- один з базових компонентів будь-якої просунутої системи. При вході по посиланню на веб-додаток в першу чергу користувач стикається з сторінкою входу в систему.

Після того, як користувач увійде в систему, він потрапляє на головну сторінку, де є можливість вибрати один з наданих модулів.

Лівий блок сайту являє собою спектр модулів(сторінок).

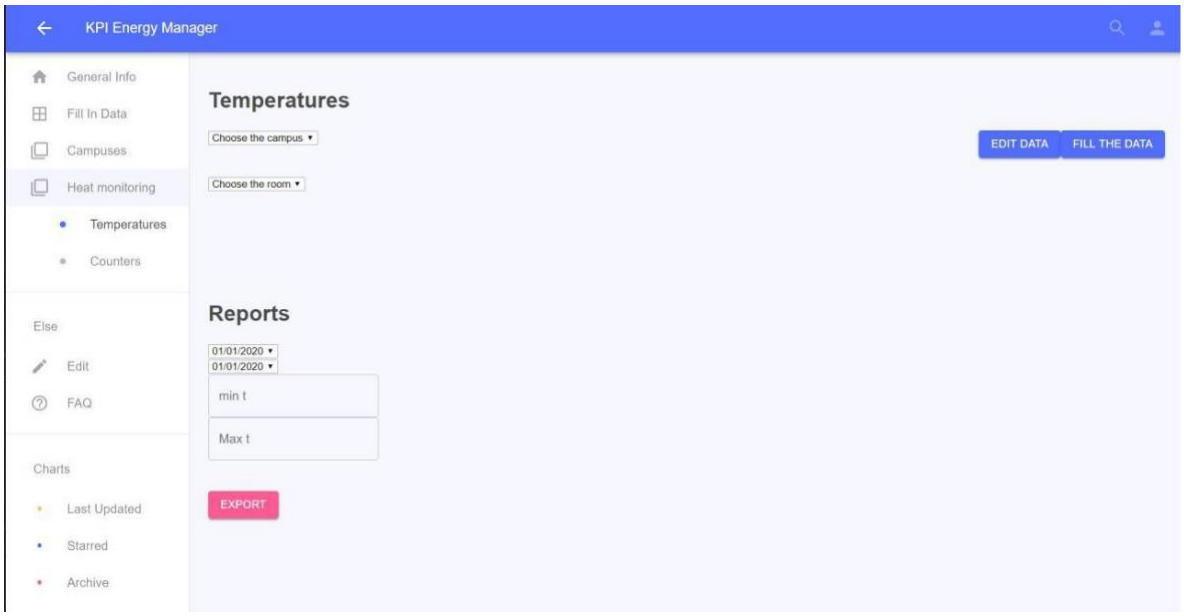
Рис. 5.2. Бічна панель з вибором модулів системи.



Будь-який користувач має можливість перейти та переглянути інформацію цих сторінок (інформація відображається відповідно до рівня доступу користувача). Обравши пункт «температури» у модулі тепло моніторингу, користувач отримує можливість експортувати дані за заданими фільтрами (дати, проміжок температур, тощо) у формат Excel.

А для вищих ж рівнів доступу буде надана можливість заповнення таблиць інформації(дата, показники і т.д.).

Рис. 5.3. Сторінка взаємодії для заповнення температуних норм.



Простий інтерфейс сторінки заповнення даних про температуру, а також інших важливих облікових одиниць даних дозволяє максимально комфортно опрацювувати систему як низькорівневому користувачеві, так і надійно і безпечно надавати доступ до окремого модуля користувачеві-енергоменеджеру.

При натисканні на кноку «Експортувати» формується звіт у форматі Excel та завантажує його користувачеві відповідно до дат, які задав користувач. У звіті доступні наступні поля: Назва корпусу, номер аудиторії, температура у приміщенні, температура ззовні, нормативна температура

для цього типу аудиторій, та відповідальна особа, що проводила вимірювання.

Рис. 5.4. Експортований файл у форматі Excel.

Дата проведення в	Корпус	№ кімнати	Час проведення в	Фактична т	Нормативна темпе	Зовнішня температура
		110	12:30:00	16		13
06.05.2020		111	19:22:21	14		12
		112	18:11:39	15		11
07.05.2020	Корпус 1	113	18:11:39	13	18	11
		110	12:30:00	14		11
08.05.2020		111	19:22:21	16		16
		112	18:11:39	15		15
09.05.2020	Корпус 1	113	18:11:39	15	18	12
		110	12:30:00	17		11
10.05.2020		111	19:22:21	18		17
		112	18:11:39	19		18
11.05.2020	Корпус 1	113	18:11:39	19	18	18

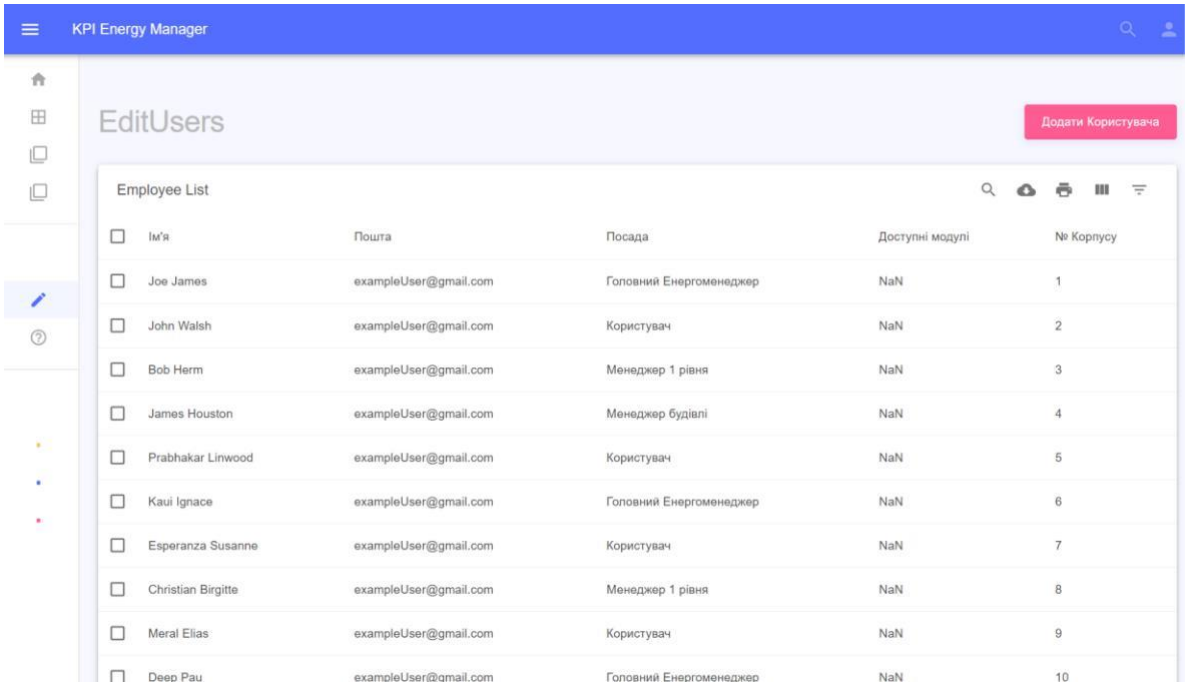
Важливим модулем у системі є модуль стоврення-редагування-управління користувацькими аккаунтами. На найвищому адміністративному рівні доступні додаткві функціональні можливості редагування та додавання користувачів з найрізноманітнішими правами доступу.

Модуль відображає загальну, важливу інформацію про користувача, та містить функції експорту списку користувачів, фільтр пошуку користувача за роллю та номером будівлі, а також пошук по доступним модулям.

Головний енергоменеджер –адміністратор з правами найвищих ієрархічних можливостей. Рівень доступу особистої сторінки з такими правами дозволяє не лише надавати можливість доступу окремому одиничному користувачу, а й додавати до його функціональних

можливостей управління тим чи іншим окремим модулем, навіть до фактичних можливостей рівня головного енергоменеджера за виключенням переліку модулів.

Рис. 5.5 Сторінка редагування списку користувачів.



Головна ж сторінка додатку містить в собі основну інформацію про будівлю, та загальну інформацію про вибрану аудиторію та відображає виведені дані що формуються у вигляді графіків та гістаграм.

Графіки та гістаграми у своїй функціональній можливості можливі у маштабувані та виборі відображення по датам до поточного числа включно.

Рис 5.6. Головна сторінка з загальною інформацією про корпус.

Автоматичне оновлення даних на сторінці проводиться опісля перезавантаження веб-застосунку або ж після вибору корпусу та кімнати по натисканню на кнопку ‘Вибрати корпус’.

Надалі проводиться аналіз даних та відображається інформація про стан обладнання та облік енергоресурсів заданих будівель.

Висновок

При розробці програмного продукту системи АРМ було проаналізовані та виділені основні аспекти енергетичного управління в автоматизованих системах вибраного напрямку, а згодом реалізовані в програмному продукті. Детально розібране поняття АРМ та енергоефективності та структурований план коректної роботи системи, що базується на автоматизованому робочому місці. Відповідно до функціоналу системи було розроблено базу даних, а також запрограмовану саму систему з точки зору архітектурного підходу у вигляді model – view – controller.

Досліджено та використано поняття клієнт-серверної розробки та створено CRUD інтерфейс користувача.

В ході виконання роботи розроблене АРМ для обробки інформації про використання енергоресурсів та температурних режимів з метою оптимізації їх використання.

Список використаних джерел

1. Аппак М.А. Автоматизированные рабочие места на основе персональных ЭВМ.- М.: Радио и связь, 1989.-176 с.: ил.
2. Learn JS [Електронний ресурс] – Режим доступу до ресурсу:https://learn.javascript.ru/.
3. Системи Енергоменеджменту [Електронний ресурс] – Режим доступу до ресурсу:https://core.ac.uk/download/pdf/48403299.pdf. .
4. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/.
5. Дзядикевич Ю. В. Енергетичний Менеджмент / Ю. В. Дзядикевич, М. В. Буряк, Р. І. Розум., 2010. – 215 с.
6. Ю.В.Хацкевич, Г.Г.Півняк. Системи енергоменеджменту та їх практичне забезпечення [Електронний ресурс] / Ю.В.Хацкевич, Г.Г.Півняк. — 2013. — 215с. .
7. ASEM [Електронний ресурс] – Режим доступу до ресурсу: https://asem.com.ua/.
8. Енергоменеджмент. Системи. [Електронний ресурс] – Режим доступу до ресурсу: https://decentralization.gov.ua/energoefect/enerhomenedzhment.
9. work with MySql [Електронний ресурс] – Режим доступу до ресурсу: http://www.sql-tutorial.ru/.
10. АРМ Енергоменеджера [Електронний ресурс] – Режим доступу до ресурсу: https://www.cenef.kiev.ua/arm_02/manre_files/frame.htm.
11. Поняття енергоефективності [Електронний ресурс] – Режим доступу до ресурсу: https://www.tovgeo.vn.ua/energy.

Додаток 1 – Специфікація

Позначення	Найменування	Примітки
Документація		
1	Пояснювальна записка	н/п
2	Додаток 2. Текст програмного модулю	н/п
3	Додаток 3. Опис програмного модулю	н/п
Компоненти		
1	Модель вигляду	Прослуховує події викликані користувачем, відслідковує реактивні дані
2	Роутинг REST API	Визначає типи запитів, передані параметри, а також вказує необхідність авторизації через middleware
3	Метод очищення даних	Задача методу полягає у виявлені та видаленні не потрібних даних з основною метою поліпшення якості даних.
4	Метод розмежування	Метод розмежування дозволяє встановлювати чіткі кордони між правами та ролями користувачів. Базується на приватних роутах(шляхах) та відповідає за управління доступу до окремих модулів.
7	Метод відображення текстових даних	Метод полягає у візуальному представленні списків відфільтрованих даних.

8	Метод редагування користувача	Метод полягає у можливості змінювати облікові статичні дані про певного користувача або ж групу користувачів. Даних модуль доступний лише для певних ієрархічних груп управління системи.

Додаток 2 – Лістинг програми

1.1. Layout.js

```
import React from "react";
import {
  Route,
  Switch,
  Redirect,
  withRouter,
} from "react-router-dom";
import classnames from "classnames";

// styles
import useStyles from "./styles";

// components
import Header from "../Header";
import Sidebar from "../Sidebar";

// pages
import Dashboard from "../../pages/dashboard";
import Maps from "../../pages/maps"; import
Tables from "../../pages/tables";
import Icons from "../../pages/icons";
import Charts from "../../pages/charts";
import ChooseCampus from '../AddCampus/ChooseCampus';
import AddCampus from '../AddCampus/AddCampus'; import
Error from '../../pages/error/Error';
import EditUsers from '../../pages/editUsers/EditUsersPanel.js';
```

```
// context
import { useLayoutState } from "../../context/LayoutContext"; import
Temperatures from "../../pages/temperatures/Temperatures"; import
Counters from "../../pages/counters/Counters";

function Layout(props) {
  var classes = useStyles();

  // global
  var layoutState = useLayoutState();

  return (
    <div className={classes.root}>
        <>
          <Header history={props.history} />
          <Sidebar />
          <div
            className={classnames(classes.content, {
              [classes.contentShift]:
            layoutState.isSidebarOpened, })}
          >
            <div className={classes.fakeToolbar}
            /> <Switch>
              <Route path="/app/dashboard" component={Dashboard} />
              <Route path="/app/tables" component={Tables} /> <Route

                exact
```

```
            path="/app/ui"
            render={() => <Redirect to="/app/ui/icons" />}
          />
          <Route path="/app/ui/maps" component={Maps} />
          <Route path="/app/ui/icons" component={Icons} />
          <Route path="/app/ui/choosecampus" component={ChooseCampus} /
>
          <Route path="/app/ui/charts" component={Charts} />
          <Route path="/app/ui/temperatures" component={Temperatures} />
          <Route path="/app/ui/counters" component={Counters} /> <Route
          path="/app/ui/addcampus" component={AddCampus} /> <Route
          path="/app/ui/error" component={Error} />
          <Route path="/app/ui/editusers" component={EditUsers}
        /> </Switch>
      </div>
    </>
  </div>
 );
}

export default withRouter(Layout);
```

1.2. AddCampus.js

```
import React from 'react';
import PageTitle from '../PageTitle/PageTitle'; //
import {API_URL} from '../../pages/login/url';

export default class AddCampus extends React.Component {
```

```
    constructor(props) {
        super(props);
        this.state = {
            number: '',
            floors_count: '',
            area: '',
            last_renewal_year: '',
        };




        this.sendData = this.sendData.bind(this);
        this.handleChange = this.handleChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
    }
    handleChange(event) {
      this.setState({
        [event.target.name]: event.target.value});
    }


    sendData() {
      console.log(this.state.number, this.state.floors_count, this.state.area,
this.stat e.last_renewal_year);
        fetch('https://f6d056320b58.ngrok.io/diplom/public/index.php/api/building
?XDEBUG_SESSION=PHPSTORM', {
          method: 'POST',
          mode: 'cors',
          headers: {
```

```
            'Content-type': 'application/json',
            'Authorization': 'Bearer ' + "DgtaPv3ciCHVUqABVrha",
            'Accepts':'application/json'
          },
           body: JSON.stringify({
            number: this.state.number,
            floors_count: this.state.floors_count,
            area: this.state.area,
            last_renewal_year: this.state.last_renewal_year,
          })
      }) /*end fetch */
      .then(results => results.json());
}



      handleSubmit(event) {
        alert('Данні про корпус №'+this.state.number+'
        збереженo'); event.preventDefault();
      }
    render() {
        return (
            <>
      <PageTitle title="Add Campus"/>
    <div>
      <form onSubmit={this.handleSubmit}>
        <label>
          Номер Корпусу
          <input type="text" name="number" onChange={this.handleChange}/>
```

```
        </label>
        <br/><br/>
        <label>
          Кількість поверхів
          <input type="text" name="floors_count"
placeholder={this.state.floors_c ount} onChange={this.handleChange}/>
        </label>
        <br/><br/>
        <label>
          Загальна Площа
          <input type="text" name="area" placeholder={this.state.area}
onChange= {this.handleChange}/>
        </label>
        <br/><br/>
        <label>
            Рік останнього ремонту
          <input type="text" name='last_renewal_year'
placeholder={this.state.last_ renewal_year} onChange={this.handleChange} />
        </label>
      <br/><br/>
      <input type="submit" value="Надіслати" onClick={this.sendData}/>
      </form>
    </div>
    </>
       )
    }
}
```

Додаток 3

АРМ енергоменеджера з обліку поточних активів.

Опис програми

Аркушів 1

Київ – 2020

АНОТАЦІЯ

Розроблений програмний продукт являє собою універсальну систему, що слугуватиме для моніторингу та обліку енергетичних ресурсів, автооновленню даних з лічильників та обробкою інформації у вигляді графічних таблиць для виведення на екран. Додаток розроблений у вигляді архітектурного патерну модель – вигляд – контролер, що забезпечує швидкість та надійність, а також серверна частина застосунку представлена у вигляді REST API, що дає простір для розробки у майбутньому також і мобільного додатку. Для розробки клієнтської частини застосунку був використаний JavaScript фреймворк React з використанням Redux бібліотеки.

У функціонал системи входить облік енергетичних ресурсів, моніторинг показників, а також відслідковування температурних режимів будівель.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ ТА ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система АРМ енергоменеджера з обліку поточних активів була розроблена з використанням мов програмування JavaScript та PHP. Для клієнтської частини був використаний фреймворк React, з використанням різноманітних бібліотек:

- Redux - відкрита JavaScript бібліотека призначена для управління станом програми. Найчастіше використовується разом з React або Angular для побудови інтерфейсів користувача.
- Axios – бібліотека для створення AJAX запитів. Полегшує роботу з запитами на сервер, дає зручні методи для встановлення хедерів, get-параметрів і інших загальних налаштувань.
- Chart.js – бібліотека для створення інтерактивних графіків. В даному випадку використовувалась надбудова vue-chartjs, яка створена для роботи з фреймворком Vue.js, та підтримує реактивність даних.
- Jest – це бібліотека JavaScript для тестування коду, що підтримується Facebook. з акцентом на простоту. Вона працює з проектами, використовуючи: Babel, TypeScript, Node.js, React, Angular та Vue.js.

Функціонал системи можна поділити на три основні складові: облік даних лічильників, моніторинг енергетичних показників, а також відслідковування температурних режимів будівлі.

Користувач відносно рівня доступу може переглядати, редагувати, видаляти дані про показники лічильників. При перегляді інфомації про дані лічильників можна вибрати інформацію за певний період від і до якоїсь дати.

І останній функціонал – це керування списками користувачів. Адміністратор може створювати нових користувачів з різними правами доступу, а також редагувати інформацію існуючих користувачів системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Програмний продукт складається з клієнтської частини та серверної частини (готовий API) з використанням патерну MVC (Model-View-Controller). Сполучення серверної і клієнтської частини відбувається через REST API (рис.1).

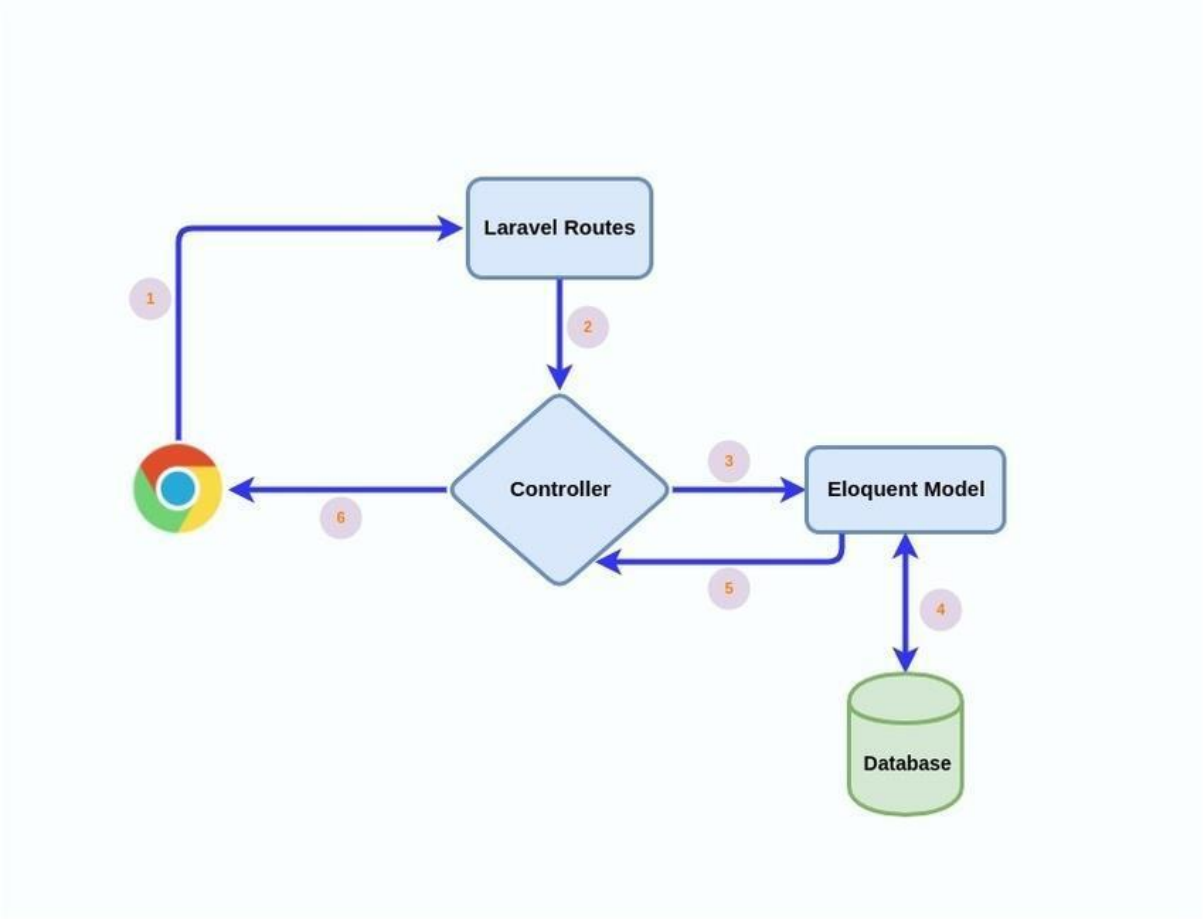


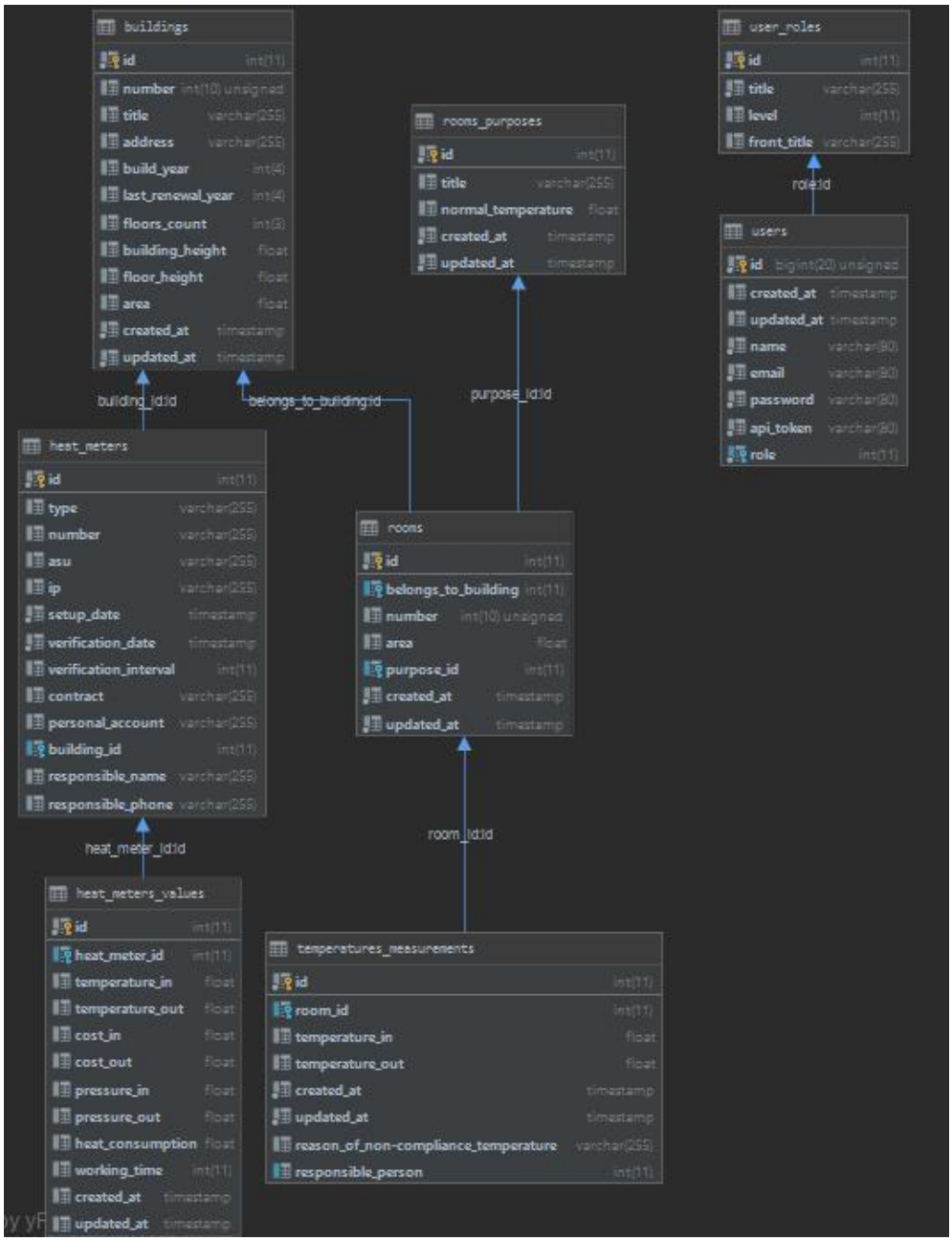
Рисунок 1 – Архітектура системи.

Дані на клієнтській частині (в нашому випадку це токен для доступу до API), ми зберігаємо в сховищі браузера LocalStorage, решта даних відповідно до архітектури, зберігаються на серверній стороні програми.

У серверної частини модель представлена класами, які відображають сутності баз даних, для кожного показника створений свій контроллер, який викликає методи показника, а виглядом є шляхи зв’язку з клієнтською частиною. Дані на серверній частині зберігаються в базі даних MySQL.

Останнім пунктом системи є MySQL базою даних. База даних представлена основними таблицями – таблицями користувачів та їх ролей, показників, будівель (рис. 2).

Рисунок 2. База даних.



ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Для розробки програмного забезпечення, а саме його клієнтської частини, було обрано наступні технології: мова програмування JavaScript, фреймворк ReactJS, постачальник модулів Webpack, менеджери пакунків Yarn та NPM та Sass — скриптова метамова, яка інтерпретується в каскадні таблиці стилів.

Також було обрано такі засоби розробки: середовище розробки WebStorm, система контролю версій Git, бібліотека для тестування Jest.

Сама система не потребує інсталяції, а запускається через браузер. Система може використовуватись на різних операційних системах. Для запуску підходять різні види браузерів, наприклад Google Chrome, Mozilla Firefox, Safari, Opera і тд.

ВХІДНІ І ВИХІДНІ ДАНІ

Фактично більшість вхідних даних надходять в систему від користувача. Користувач, в ході використання системи заповнює інформацію про енергетичні показники, температурні режими та будівлі з клавіатури. Деякі поля пропонують автозаповнення. Інформація, що вводяться користувачем з клавіатури валідуються на стороні сервера та на стороні клієнта. Можна вважати що вхідними данними є створення нових корпусів та прилеглих до них аудиторій, що створюються користувачем з певними ієрархічними можливостями.

Другим типом вхідних даних, є дані, що експортуються з Exel-таблиць. Інформація, в такому разі, одразу відправляється та обробляється на серверній частині системи.

Вихідними даними є списки корпусів та аудиторій, таблиці користувачів, загальна інформація про стан та останній рік ремонту будівлі, графіки та таблиці з показниками енергоресурсів та показники температурних вимірів ззовні та всередині будівель.